Journal of Computational Physics **153,** 509–534 (1999)
Article ID jcph.1999.6293, available online at http://www.idealibrary.com on IDEAL®

An Adaptive Version of the Immersed Boundary Method

Alexandre M. Roma,* Charles S. Peskin,† and Marsha J. Berger†

*Instituto de Matemática e Estatística, Universidade de São Paulo, Caixa Postal 66281, São Paulo, SP 05315-970, Brazil; and †Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012
E-mail: roma@ime.usp.br, peskin@cims.nyu.edu, berger@cims.nyu.edu

Received July 13, 1998; revised February 24, 1999

A computational setting for the Immersed Boundary Method employing an adaptive mesh refinement is presented. Enhanced accuracy for the method is attained locally by covering an immersed boundary vicinity with a sequence of nested, progressively finer rectangular grid patches which dynamically follow the immersed boundary motion. The set of equations describing the interaction between a non-stationary, viscous incompressible fluid and an immersed elastic boundary is solved by coupling a projection method, specially designed for locally refined meshes, to an implicit formulation of the Immersed Boundary Method. The main contributions of this work concern the formulation and the implementation of a multilevel self-adaptive version of the Immersed Boundary Method on locally refined meshes. This approach is tested for a particular two-dimensional model problem, for which *no significant difference* is found between the solutions obtained on a mesh refined locally around the immersed boundary, and on the associated uniform mesh, built with the resolution of the *finest level*.

Key Words: incompressible flows; interface problems; mesh refinement; projection methods; immersed boundary method.

1. INTRODUCTION

Many problems in biofluid dynamics involve the interaction between a non-stationary, incompressible viscous fluid and a visco-elastic biological tissue, which may have time-dependent configuration, time-dependent elastic properties, or both (e.g., the interaction between blood, heart muscles, and heart valve leaflets). Although these problems can be handled in a robust manner by the *Immersed Boundary Method* and qualitatively good results be obtained [10, 14, 15, 27–29, 32, 35], this method suffers from a certain "lack of resolution." Thin boundary layers, which usually develop along the biological tissue, and fine

0021-9991/99 $30.00

geometrical details can be adequately resolved only if the computational mesh is very fine. If a uniform mesh is used, this requirement is inevitably extended to the entire computational domain, and the resulting mesh may exceed the storage capacity of the computer.

This accuracy issue introduced by localized phenomena around some kind of interface is common to many problems, arising in many different fields. In the past few years, much effort was spent to have this issue appropriately addressed and, as a result, several *adaptive methods* were introduced.

Employing a very particular approach, Li [22, 23] and LeVeque and Li [20, 21] developed the *Immersed Interface Method*, designed to solve problems with non-smooth solution across interfaces. The method adapts the finite difference scheme in a neighborhood of the interface to obtain an equally accurate solution at all points on a uniform Cartesian grid. The main idea is to incorporate the known jumps in the solution or in its derivatives across the interface into the scheme, obtaining a modified scheme close to the interface.

Level set methods, in combination with adaptive mesh refinements, were considered by Haj-Hariri, Shi, and Borhan [17], who studied the three-dimensional motion of deformable viscous drops, and by Sussman *et al.* [36, 37], who studied incompressible two-phase flows with surface tension, two-dimensional axisymmetric and fully three-dimensional air bubbles and water drops.

In the context of gas dynamics, Bayyuk, Powell, and van Leer [3] introduced a method for performing simulations of Euler flows around moving and deforming bodies in two dimensions; their method employed Cartesian, unstructured, quadtree-based grids and finite-volume conservative discretization. Greenough *et al.* [16] presented an interface-capturing method coupled to a local mesh adaptive refinement for solving compressible multifluid equations in complex geometries.

More recently, in the context of unsteady, incompressible flows, Agresar [1], and Agresar *et al.* [2] performed simulations of moving and deforming circulating cells, tracking expicitly cell interfaces employing the Euler–Lagrangian method developed by Unverdi and Tryggvason [39], while solving the fluid equations on the adaptive, unstructured Cartesian grids of Bayyuk, Powell, and van Leer [3].

Still in the context of unsteady, incompressible flows, Roma [34] introduced a new computational setting for the Immersed Boundary Method employing a hierarchical, nested adaptive mesh refinement [6–9, 33]. In this approach, accuracy enhancement can be achieved by covering locally an immersed boundary vicinity with a sequence of nested, progressively finer rectangular grid patches which dynamically follows the immersed boundary motion.

Here, this *adaptive version* of the Immersed Boundary Method is presented, and a more efficient dynamical strategy for recomputing the locally refined meshes introduced. Section 2 presents the fluid–interface interaction equations for the model problem to be considered. Section 4 explains how locally refined grids are generated and updated, while Sections 3, 5, and 6 present the discretizations of the equations in time and space on these grids. Section 7 highlights the numerical scheme employed to solve the discretized set of equations for the specific model problem considered, and Sections 8 and 9 present the numerical results and the conclusions, respectively.

2. STATEMENT OF THE PROBLEM

Consider a two-dimensional incompressible fluid which contains an immersed boundary (immersed elastic interface supposed to be infinitely thin and massless). Using the

Immersed Boundary Method formulation, one can write the equations of motion of the system composed of the fluid and the immersed boundary as

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\nabla p}{\rho} = \frac{\mu}{\rho}\Delta \mathbf{u} - \mathbf{u}\cdot\nabla \mathbf{u} + \frac{\mathbf{F}}{\rho} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where

$$\mathbf{F}(\mathbf{x}, t) = \int_S \mathbf{f}(s, t)\, \delta(\mathbf{x} - \mathbf{X}(s, t))\, ds \tag{3}$$

$$\mathbf{f}(s, t) = \frac{\partial (T\boldsymbol{\tau})}{\partial s} \tag{4}$$

with

$$\frac{\partial \mathbf{X}(s, t)}{\partial t} = \int_\Omega \mathbf{u}(\mathbf{x}, t)\, \delta(\mathbf{x} - \mathbf{X}(s, t))\, d\mathbf{x}. \tag{5}$$

In (1)–(2), the physical parameters ρ and μ, assumed to be constants, are respectively the mass density and the viscous coefficient of the fluid, p is the hydrodynamical pressure, $\mathbf{u}$ is the fluid velocity, and $\mathbf{F}$, which usually would be an external force field, in this context represents the singular elastic force distribution (3), differing from zero only on the immersed boundary points $\mathbf{X}(s, t)$, where $\delta(\cdot)$ is the two-dimensional Dirac delta; p, $\mathbf{u}$, and $\mathbf{F}$ are functions of the time t and of the *Eulerian* coordinate $\mathbf{x}$, which is defined on a rectangular domain Ω.

The elastic force density $\mathbf{f}(s, t)$ in (4), defined along the immersed boundary, is a function of the unit tangent to the immersed boundary, $\boldsymbol{\tau}$,

$$\boldsymbol{\tau} = \frac{\partial \mathbf{X}/\partial s}{\|\partial \mathbf{X}/\partial s\|}, \tag{6}$$

and of the immersed boundary tension T, which will be introduced for a special case below. Although the derivation of (4) will not be included here, it can be found in [28].

The shape of the immersed boundary may be quite complicated and its motion not known in advance. Since the elastic force distribution $\mathbf{F}$ is computed from the configuration of the immersed boundary, the position of each one of its points $\mathbf{X}(s, t)$ must be tracked in a *Lagrangian* fashion, with $s \in S$ the Lagrangian parameter. As a result of the fluid viscosity, one has (5), which states that the immersed boundary points move at the local fluid velocity.

The set of equations (1)–(6) is given in a mixed Euler–Lagrangian formulation. As a special case, consider the model problem given by a simple, closed curve for which

$$T(s, t) = T_0 \left\| \frac{\partial \mathbf{X}}{\partial s}(s, t) \right\|, \tag{7}$$

where T_0 is a non-negative constant, and $s \in S$, $S = [0, 2\pi]$, with the point $s = 0$ identified with the point $s = 2\pi$.

Note that, in order to have the problem completely posed, one has to provide an initial condition for the system $(\mathbf{X}, \mathbf{u})$ and a boundary condition for $\mathbf{u}$. To simplify the problem,

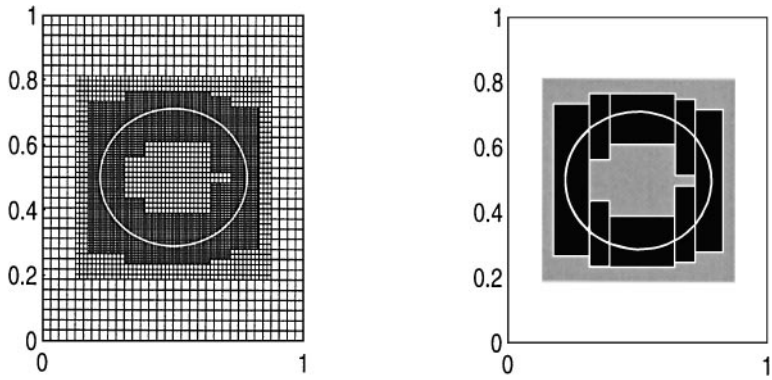

FIG. 1. Grid lines (left) and grid patches (right).

periodic boundary conditions for **u** will be adopted. The initial state of the system will be provided later.

If T_0 is equal to zero in (7), the immersed boundary will not be elastic and it will be carried passively by the fluid. On the other hand, if T_0 is greater than zero, the problem is the two-dimensional analog of an elastic spherical balloon, filled with the same fluid present outside, whose motion is driven by the elastic force acting on its wall. In this case, the immersed boundary tends to shrink to a point but it is prevented from doing so by the incompressibility of the fluid. Also, because of the fluid incompressibility, the immersed boundary equilibrium configuration is a circle that encloses the same area as its initial configuration.

Considering (4) and (6)–(7), one can write the elastic force density for this model problem as

$$\mathbf{f}(s,t) = T_0 \frac{\partial^2 \mathbf{X}(s,t)}{\partial s^2}. \tag{8}$$

Equations (1)–(5) can be naturally divided into two groups: one, the *Navier–Stokes equations* (1)–(2), describing the motion of the incompressible viscous fluid, and the other one, the *fluid-boundary interaction equations* (3)–(5), describing the interaction between the fluid and the immersed boundary. The next sections present a discretized form of these equations for the model problem introduced, which will be solved numerically on *composite grids* like the one in the Fig. 1.

3. DISCRETIZATION IN TIME

3.1. *Navier–Stokes Equations*

The numerical solution of the Navier–Stokes equations on composite grids will be computed by a projection method inspired by the projection method introduced by Bell, Colella, and Glaz [5], and by the approximate projection on locally refined grids developed by Minion [25], both second-order variations of Chorin's original projection method [11, 12].

The temporal discretization of the Navier–Stokes equations (1)–(2) is based on a Crank–Nicholson type of scheme where first the advection–diffusion equation is solved to obtain a

provisional velocity field, which is then projected onto the space of discretely divergence-free vector fields (up to a given convergence tolerance).

The starting point is the equations

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}+\frac{\nabla p^{n+\frac{1}{2}}}{\rho}=\frac{\mu}{\rho}\Delta\left(\frac{\mathbf{u}^{n+1}+\mathbf{u}^n}{2}\right)-[(\mathbf{u}\cdot\nabla)\mathbf{u}]^{n+\frac{1}{2}}+\frac{\mathbf{F}^{n+\frac{1}{2}}}{\rho} \tag{9}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0. \tag{10}$$

The superscripts denote instants of time, e.g., $\mathbf{u}^n \doteq \mathbf{u}(\cdot, t^n)$, $\mathbf{F}^{n+1/2} \doteq \mathbf{F}(\cdot, t^{n+1/2})$, where $t^n \doteq t_0 + n\Delta t$ and $t^{n+1/2} \doteq t_0 + (n+\frac{1}{2})\Delta t$. To simplify the problem, only boundary conditions of periodic type are being considered for the velocity $\mathbf{u}$.

Usually, in the Crank–Nicholson scheme the nonlinear advection term at $t^{n+1/2}$ would be obtained by the average of its value at the times t^n and t^{n+1}. Bell, Colella, and Glaz though, in their second-order projection method [5], introduced an *explicit* computation of this term employing an upwind strategy based on the Godunov methodology developed by Colella [13] which is appropriate for flow at high Reynolds number. Only known values of the velocity $\mathbf{u}$, the forcing term $\mathbf{F}$, and the approximation to ∇p from the previous time step are required. Here, for simplicity, this upwind strategy will be replaced by a quite "standard" second-order spatial discretization. This simpler approximation will be adequate at moderately high Reynolds number flows which are in sight in future applications (up to 500 or so, e.g., as in the blood flow in the heart chambers). The detailed computation of the nonlinear advection term is a subject considered in Section 6. For the moment, this term is assumed to be known.

Instead of trying to solve Eqs. (9)–(10) directly, we use the following iterative approach,

$$\frac{\mathbf{u}^{*,m}-\mathbf{u}^n}{\Delta t}+\frac{\nabla p^{n+\frac{1}{2},m-1}}{\rho}=\frac{\mu}{\rho}\Delta\left(\frac{\mathbf{u}^{*,m}+\mathbf{u}^n}{2}\right)-[(\mathbf{u}\cdot\nabla)\mathbf{u}]^{n+\frac{1}{2}}+\frac{\mathbf{F}^{n+\frac{1}{2},m-1}}{\rho} \tag{11}$$

$$\frac{\mathbf{u}^{n+1,m}-\mathbf{u}^n}{\Delta t}+\frac{\nabla p^{n+\frac{1}{2},m}}{\rho}=\frac{\mathbf{u}^{*,m}-\mathbf{u}^n}{\Delta t}+\frac{\nabla p^{n+\frac{1}{2},m-1}}{\rho} \tag{12}$$

$$\nabla\cdot\mathbf{u}^{n+1,m}=0, \tag{13}$$

where m is the iteration number. Note that (11) is an implicit system to be solved for $\mathbf{u}^{*,m}$, for which no constraint of incompressibility is imposed. Moreover, the pressure used is known from the previous iteration. Also, note that the right-hand side of (12) is the left-hand side of (11).

For the particular problem considered, the evaluation of the force $\mathbf{F}$ will depend on the unknown value of the velocity $\mathbf{u}^{n+1}$ and, hence, it will depend on the iteration number in a manner that will become clear in the next section.

3.2. *Fluid–Boundary Interaction Equations*

Since its introduction by Peskin [26, 27] a serious issue of numerical stability has haunted the Immersed Boundary Method, whose origin is the stiffness introduced into the problem through the elastic properties of the immersed boundary.

The elastic force distribution (3) can be computed from $\mathbf{X}^n$, the boundary configuration at the beginning of the step from time level n to time level $n+1$. But, if the boundary is too

stiff or if the time step is too large, this *explicit* formulation of the method typically leads to "explosively" unstable results. One way of improving the stability of the method is to compute the elastic force not from $\mathbf{X}^n$ but from $\mathbf{X}^*$, *an estimate* of $\mathbf{X}^{n+1}$, the configuration of the immersed boundary at the end of that time step. This is referred to as the *approximately implicit* formulation of the Immersed Boundary Method and was first introduced in [26]. This formulation improves, but does not completely eliminate, the stability problems of the method.

Tu and Peskin [38] compared the stability properties displayed by three different formulations of the Immersed Boundary Method for a steady Stokes flow. They used the explicit, the approximately implicit, and a third formulation, which was *implicit in the computation of the elastic force density* $\mathbf{f}(s,t)$; that is, the elastic force density was computed from the boundary configuration at the end of the time step, $\mathbf{X}^{n+1}$, but applied to the fluid at the boundary configuration $\mathbf{X}^n$ corresponding to the beginning of the time step (in a *fully implicit* scheme, $\mathbf{X}^{n+1}$ would be used for both purposes). Significantly more expensive than the others, this last implementation presented excellent stability properties, seeming to be unconditionally stable for the model problem used.

In the context of the full Navier–Stokes equations, Mayo and Peskin [24] introduced two implicit schemes for the Immersed Boundary Method: one implicit only in the elastic forces, as in the previous work done by Tu and Peskin mentioned above, and the other one fully implicit. The former, despite having a region of stability greater than the approximately implicit scheme, still was not unconditionally stable. On the other hand, the fully implicit scheme presented was always able to preserve the stability. In all cases, the model problem used was the two-dimensional analog of an elastic spherical balloon, described in Section 1.

Here, yet *another* fully implicit scheme is presented. The Crank–Nicholson scheme (9)–(10) used in the time discretization of the Navier–Stokes equations requires that the forcing term, appearing in the momentum equations (9), be computed at half-time level. If the average of $\mathbf{F}^n$ and $\mathbf{F}^{n+1}$ is used then a *new* implicit form of the Immersed Boundary Method can be proposed. To (9)–(10) one adds

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} = \frac{1}{2}\int_\Omega [\mathbf{u}^n \delta(\mathbf{x}-\mathbf{X}^n) + \mathbf{u}^{n+1}\delta(\mathbf{x}-\mathbf{X}^{n+1})]\,d\mathbf{x} \tag{14}$$

$$\mathbf{F}^{n+1} = T_0 \int_0^{2\pi} \frac{\partial^2 \mathbf{X}^{n+1}}{\partial s^2}\,\delta(\mathbf{x}-\mathbf{X}^{n+1})\,ds, \tag{15}$$

and then $\mathbf{F}^{n+1/2}$ is defined as

$$\mathbf{F}^{n+\frac{1}{2}} = \frac{1}{2}(\mathbf{F}^n + \mathbf{F}^{n+1}), \tag{16}$$

where $\mathbf{F}^n$ is obtained simply by replacing $n+1$ by n in (15).

These equations, together with Eqs. (9)–(10), define a non-linear system for the unknown boundary configuration $\mathbf{X}^{n+1}$ along with the unknown fluid velocity $\mathbf{u}^{n+1}$, which will be solved numerically by an iterative scheme defined in Section 7.

4. COMPOSITE GRID DESCRIPTION, GENERATION AND REGRIDDING

Peskin and McQueen [28] concluded that the lack of resolution of the Immersed Boundary Method has its origin from local phenomena taking place in a neighborhood of the

immersed boundary (e.g., boundary layers and singular forces). Roughly speaking, by applying the method with a second-order fluid solver to a three-dimensional model problem, they observed through numerical convergence analysis that the fluid solver exhibited a clear second-order behavior away from the immersed boundary, the behavior being only first-order when the fluid mesh points close to the immersed boundary were considered in the analysis.

The existence of local phenomena and the need for more grid points to capture the finer geometric details of the immersed boundary suggest a possible remedy to the problem: the application of a *local mesh refinement technique*. Employing the composite grids described by Berger and Colella in [7], refined regions will be covered by a hierarchical sequence of nested, progressively finer levels $l = 1, 2, \ldots, l_{\text{finest}}$. Each level l is formed by a set of disjoint rectangular grids $G_{l,k}$, $k = 1, 2, \ldots, n_l$, that is,

$$\{\text{level } l\} = \bigcup_k G_{l,k},$$

with $G_{l,j} \cap G_{l,k} = \emptyset$, $j \neq k$ (two different grids in the same level do not overlap!), which have the same mesh spacing h_l, and whose sides are aligned in the coordinate directions. As an example, $\{\text{level } 1\} = G_{1,1}$, where $G_{1,1}$ is a global uniform grid covering Ω, the rectangular domain used in the model problem considered.

Although in gas dynamics problems refinement in time comes naturally along with refinement in space, this will not be the approach employed in this work. No time refinement will be used. All grids, in all levels, will evolve together in time, with the time step of the finest level. In the incompressible case, there is not a finite limit to the speed at which disturbances can propagate in the flow. Since each part of the incompressible flow influences all other parts instantaneously, it is not clear how different time steps could be used on the different grids.

Grids at different levels in the grid hierarchy must be "properly nested." This means that they must satisfy the following two properties:

1. a fine grid starts and ends at the corner of a cell in the next coarser level;
2. there must be at least one level $(l-1)$ cell in some level $(l-1)$ grid separating a grid cell at level l from a cell at level $(l-2)$, in the north, south, east, and west directions.

Figure 2 shows one grid at level 3, $G_{3,1}$, two grids at level 2, $G_{2,1}$ and $G_{2,2}$, all of them laid on the underlying uniform grid $G_{1,1}$, which covers the domain completely.

Typically, the setup of a problem to be solved by the Immersed Boundary Method involves the formulation of the problem on a "physical" rectangular domain, conveniently sized and made periodic in all the directions. This domain can be discretized using the composite grids described. Level 1 is obtained through the uniform division of the domain into a regular array of computational cells, whose width and height will be assumed to be the same, h_1, for simplicity. Finer rectangular grid patches forming level l, $2 \leq l \leq l_{\text{finest}}$, have mesh spacing

$$h_l = \frac{h_1}{r^{l-1}}, \tag{17}$$

where r is the *refinement ratio* used (in general, $r = 2$ or $r = 4$).

Generation of the composite grids depends on the *flagging step*, that is, determining first the cells whose collection gives the region where refinement is to be applied. Throughout

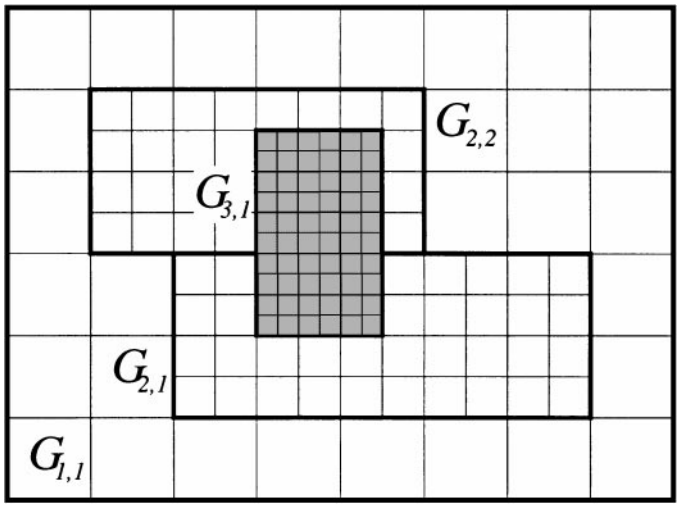

FIG. 2. Grid $G_{3,1}$ spans two coarser grids but it is properly nested.

this work, composite grids will be assumed to completely cover the immersed boundary with the finest level. Refinement levels are generated one at a time, starting with the finest level, l_{finest}, and ending with level 2. The uniform grid at level 1 is always the same and the total number of levels to be used is decided in advance.

A refinement level l is always generated by flagging cells in the next coarser level, level $(l-1)$. Flagged cells include not only cells in a vicinity of the immersed boundary but also those needed to ensure that finer grids at level $(l+1)$, if any, will be properly nested in level l. In summary, the collection of level $(l-1)$ flagged cells is obtained by the two-step procedure:

1. flag enough cells in a vicinity of the immersed boundary to guarantee that the discrete delta function adopted will have its support completely contained by grids at the level under construction;
2. to this collection of cells, add all the cells needed to have level $(l+1)$ grids properly nested in level l.

Once the collection of flagged cells is obtained, the grids which will belong to level l are generated through the application of the algorithm for point clustering developed by Berger and Rigoutsos [9], which combines elements of both computer vision and pattern recognition theory. The algorithm returns a set of non-overlapping rectangular patches for a given collection of flagged cells, finding the "best" places to cut using *signatures*. More precisely, vertical and horizontal signatures of a function $f(x, y)$ are defined respectively as

$$H(x) = \int_y f(x, y)\, dy$$

$$V(y) = \int_x f(x, y)\, dx.$$

If the function $f(x, y)$ is understood as a binary function which assumes the value 1 at the flagged cells, and 0 at the other cells, the key idea of the algorithm is to look for the

zero crossings in the second derivative of the signatures $H(x)$ and $V(y)$ (inflection points), since they detect the transitions from flagged to non-flagged regions. The best place to introduce an edge is indicated by the most "prominent" inflection point. For details, the interested reader should refer to [9]. This procedure is performed for each level until all the grid patches have been generated.

Composite grid regridding is performed always when the distance from an immersed boundary point to the border of the finest level is less than an allowed minimal distance. This minimal distance depends on the size of the support of the discrete delta function.

Several composite grids may be needed during the resolution of a problem. Their number depends essentially on the amplitude of the immersed boundary motion. For problems where its motion is highly localized, for example, few composite grids will have to be generated.

5. DISCRETIZATION IN SPACE

5.1. *Location of the Physical Variables*

Physical variables $\mathbf{u} = (u, v)$ and p are placed in a MAC staggered grid fashion (Harlow and Welch [18]). In this discretization, the cells covering the domain are considered "mass control volumes" where scalar quantities (e.g., pressure, divergence) are defined at the cell centers and vector quantities (e.g., velocity, forcing terms, pressure gradient) have their vertical component defined at the middle of the horizontal cell edges and their horizontal component defined at the middle of the vertical cell edges. At this point, a note on the convention regarding the spatial indexing of scalar and vectorial quantities seems to be appropriate. Given a computational cell (i, j), its left and right midedges will have as indices $(i - \frac{1}{2}, j)$ and $(i + \frac{1}{2}, j)$ respectively, and its top and bottom midedges $(i, j + \frac{1}{2})$ and $(i, j - \frac{1}{2})$. Velocity $\mathbf{u}_{i,j}$ on the cell (i, j) will be defined by

$$\mathbf{u}_{i,j} \doteq \left(u_{i-\frac{1}{2},j}, v_{i,j-\frac{1}{2}}\right),$$

with the horizontal component defined at the middle of the left cell edge and the vertical component defined at the middle of the bottom cell edge. Although it is arbitrary to associate (i, j) with $(i - \frac{1}{2}, j)$ and $(i, j - \frac{1}{2})$ in this manner, some such notation is needed if one is to define vector quantities at all. Since scalar quantities are defined at the cell centers, their indices are simply those defining the cell. It will be also convenient to define auxiliary computational cells underneath fine grid cells as shown in Fig. 3.

On uniform grids, the MAC location of physical variables has been used for a long time and has well understood properties. The main reason for its choice in this context was that away from coarse–fine interfaces, it is possible to define conveniently second-order approximations for the gradient and for the divergence operators, such that the discretization of the projection operator presents excellent numerical properties. It is also possible to adapt *multigrid methods* for the solution of the pressure Poisson equation resulting from substituting (13) into (12).

The disadvantages of this approach include the fact that it is not clear how to obtain higher order discretizations of the nonlinear advection terms preserving the numerical stability in regions of steep gradients without introducing either nonphysical oscillations or unnecessary dissipation. This is especially true for flows at high Reynolds numbers. If all the variables were placed at the cell centers, for example, variations of the Godunov method could be employed more "naturally" [5, 4, 19, 25].

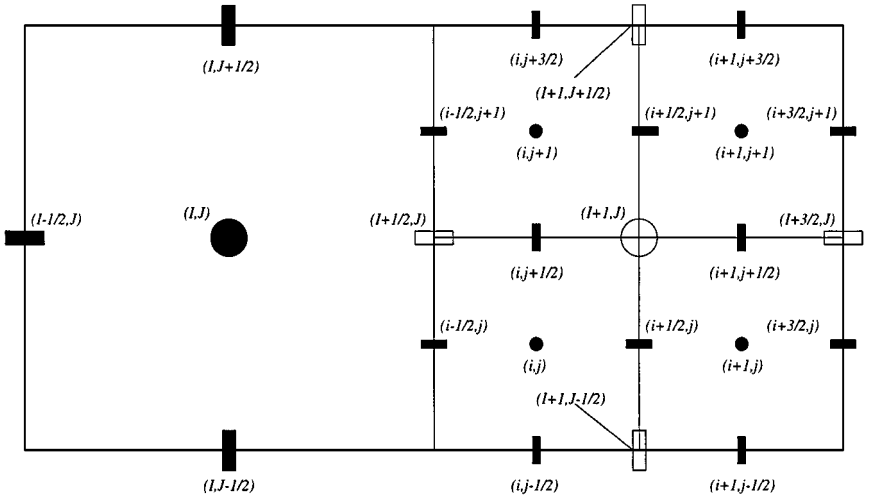

FIG. 3. Index location and coarse computational cell underneath finer grid cells.

5.2. *Divergence, Gradient, and Laplacian Difference Operators*

Given velocity and pressure located as explained above, the divergence and gradient operators to be used in (11)–(13) are given by

$$(D\mathbf{u})_{i,j} = \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{h} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{h} \tag{18}$$

$$(Gp)_{i,j} = \left(\frac{p_{i,j} - p_{i-1,j}}{h}, \frac{p_{i,j} - p_{i,j-1}}{h} \right), \tag{19}$$

where the time indices were suppressed for clarity. The discretization of the viscous terms in (11) will be given by the five-point stencil

$$(L\mathbf{u})_{i,j} = \left(\frac{u_{i+\frac{1}{2},j} + u_{i-\frac{3}{2},j} + u_{i-\frac{1}{2},j+1} + u_{i-\frac{1}{2},j-1} - 4u_{i-\frac{1}{2},j}}{h^2}, \right.$$
$$\left. \frac{v_{i,j+\frac{1}{2}} + v_{i,j-\frac{3}{2}} + v_{i+1,j-\frac{1}{2}} + v_{i-1,j-\frac{1}{2}} - 4v_{i,j-\frac{1}{2}}}{h^2} \right), \tag{20}$$

which will be also denoted as $(L\mathbf{u})_{i,j} = ((Lu)_{i-1/2,j}, (Lv)_{i,j-1/2})$. The difference operators (18)–(20) are clearly second order away from coarse–fine grid interfaces.

To prevent these operators from being formally redefined at grid borders, an additional layer of cells is appended around each grid to provide boundary values. These cells are commonly called *ghost cells* and their existence is only for programming purposes. The values of $\mathbf{u}$ and p at a ghost cell are provided in a manner that depends on whether or not it coincides with a regular cell belonging to another grid in the same level. If there is such coincidence, ghost cell values are obtained by simply "importing" them from the sibling grid, a process often referred to as *injection*. On the other hand, if there is not a sibling grid to provide these values, an interpolation procedure involving values from coarse and fine grids is employed. Usually, quadratic polynomial interpolation schemes are used to provide third-order approximate values at the ghost cells. This guarantees second-order first derivatives but only first-order second derivatives along coarse–fine grid interfaces.

Nevertheless, when the flow is smooth near these interfaces, it has been observed that this is enough to furnish a second-order accurate scheme (in L_2-norm) [19]. Ghost cells adjacent to level 1 are set up only through injection by enforcing the periodic boundary conditions.

Finally, note that difference operators will be also formally kept unchanged on coarser grids by using the coarse computational cells defined underneath fine grid patches. Their values are defined through interpolations from finer values. Details of all the interpolation schemes can be found in [34].

5.3. *Discrete Dirac Delta*

In order to provide the spatial discretization of Eqs. (14)–(15) it is necessary to furnish a two-dimensional discrete form for Dirac's delta function, which will connect the Eulerian formulation used for the fluid with the Lagrangian formulation used for the immersed boundary.

The two-dimensional approximation to the delta function is given by the product

$$\delta_h^2(\mathbf{x} - \mathbf{x}_0) = \delta_h^1(x - x_0)\,\delta_h^1(y - y_0),$$

where

$$\delta_h^1(x - x_0) = \frac{1}{h}\phi\left(\frac{x - x_0}{h}\right), \tag{21}$$

is an approximation for the one-dimensional delta function, with ϕ the continuous function

$$\phi(r) = \begin{cases} \frac{1}{6}(5 - 3|r| - \sqrt{-3(1 - |r|)^2 + 1}), & 0.5 \le |r| \le 1.5 \\ \frac{1}{3}(1 + \sqrt{-3r^2 + 1}), & |r| \le 0.5 \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

where $r = (x - x_0)/h$. The function ϕ was not chosen arbitrarily. Actually, it is determined by requiring that a certain set of properties be satisfied by the discrete version of Dirac's delta function. In particular, the set of properties used to determine this approximation were:

1. $\phi(r)$ is continuous for all real numbers r;
2. $\phi(r) = 0,\ |r| \ge 1.5$;
3. $\sum_i \phi(r - i) = 1,\ \forall r$;
4. $\sum_i (r - i)\phi(r - i) = 0,\ \forall r$, and
5. $\sum_i [\phi(r - i)]^2 = \frac{1}{2},\ \forall r$,

where all the sums are performed for the integers i, such that $-\infty < i < +\infty$.

A novel feature of the delta function defined above is that the size of its support is three meshwidths in each space direction. Previous immersed boundary computations have used delta functions whose support was four meshwidths in each direction.

Delta functions previously employed [26, 30] obeyed a list of properties much like those above but with certain differences. In both [26] and [30], Property 3 was stronger: It required that the corresponding sums over i odd and over i even each be equal to $1/2$. The reason for this, and why we can do without it here, will be discussed below. In [26], Property 4 was not used. This led naturally (albeit non-uniquely) to a cosine shaped delta function with

a support of four meshwidths in each space direction. In [30], Property 4 was added, and the stronger form of Property 3 was still retained. This led to a uniquely determined delta function that was remarkably close quantitatively to the cosine shaped delta function, even though the cosine function does not satisy Property 4 exactly. Here we keep Property 4, but achieve a delta function with smaller support by using Property 3 in the (weaker) form stated above.

The reason why the stronger form of Property 3 was needed in previous work is that the Navier–Stokes equations were discretized employing gradient and divergence difference operators whose composition led to what looks like a "stretched" version of the usual five-point stencil for the Laplacian. Roughly speaking, as a consequence, four independent sets of equations had to be solved in the projection step (see [19] for further comments). When that happens, the stronger form of Property 3 must be imposed so that each set of equations receives equivalent contributions coming from the elastic force, with more grid points needed to satisfy all requirements. In this case, the derivation leads to a delta function which is usually four-cell supported [30]. The new, three-cell supported delta function became possible because gradient and divergence operators were discretized on MAC staggered grids. This approach naturally avoids any decoupling of equations in the projection step, making no extra property necessary.

The five properties above uniquely determine the function ϕ and hence δ_h^1. Remarkably, ϕ turns out to have a continuous derivative, though this condition was not explicitly imposed. By Property 1 above, there will be no "jumps" in the interpolation step (14) and in the *spreading step* (15) (interpolation of the velocity to and spreading of the force from immersed boundary points). Property 2 guarantees that the discrete delta function has finite support, three cells wide in this case.

In the force-spreading operation, Property 3 guarantees conservation of momentum. Properties 3 and 4 together guarantee conservation of angular momentum, and that the interpolation of linear functions will be exact, that is, smooth functions are interpolated to second-order accuracy.

Property 5 arises from considering how the force due to an immersed boundary point influences the motion of that same point, and from requiring that this influence be the same regardless of the position of the immersed boundary point relative to the mesh. Further explanations are found in [26, 30]. Note that the constant $1/2$ in the fifth property is not arbitrary. It can actually be obtained by setting $r = 0.5$ and by performing some algebraic manipulations on the set of properties. Similarly, in the condition $\phi(r) = 0$ for $|r| \geq 1.5$, the constant 1.5 is not arbitrary. It is the smallest constant consistent with the other conditions.

Finally, it is important to remember that the immersed boundary is covered completely by grids in the finest level, which is made large enough to contain the entire support of the discrete Dirac delta.

5.4. *Fluid–Boundary Interaction Equations*

The full discretization of the fluid–boundary interaction equations (14)–(15) is given by

$$\frac{\mathbf{X}_k^{n+1} - \mathbf{X}_k^n}{\Delta t} = \frac{h^2}{2} \sum_{i,j} \begin{bmatrix} u^n_{i-\frac{1}{2},j}\delta_h^2\left(\mathbf{x}_{i-\frac{1}{2},j} - \mathbf{X}_k^n\right) & +u^{n+1}_{i-\frac{1}{2},j}\delta_h^2\left(\mathbf{x}_{i-\frac{1}{2},j} - \mathbf{X}_k^{n+1}\right) \\ v^n_{i,j-\frac{1}{2}}\delta_h^2\left(\mathbf{x}_{i,j-\frac{1}{2}} - \mathbf{X}_k^n\right) & +v^{n+1}_{i,j-\frac{1}{2}}\delta_h^2\left(\mathbf{x}_{i,j-\frac{1}{2}} - \mathbf{X}_k^{n+1}\right) \end{bmatrix} \tag{23}$$

$$\mathbf{F}_{i,j}^{n+1} = T_0 \Delta s \sum_{k=1}^{N_B} \begin{bmatrix} \left(D_s^+ D_s^- X_k^{n+1}\right) \delta_h^2 \left(\mathbf{x}_{i-\frac{1}{2},j} - \mathbf{X}_k^{n+1}\right) \\ \left(D_s^+ D_s^- Y_k^{n+1}\right) \delta_h^2 \left(\mathbf{x}_{i,j-\frac{1}{2}} - \mathbf{X}_k^{n+1}\right) \end{bmatrix}, \tag{24}$$

where $\Delta s = L_B/N_B$, L_B and N_B are the length and the number of discretization points of the immersed boundary, respectively, and

$$(D_s^+ \mathbf{X})(s) = \frac{\mathbf{X}(s + \Delta s) - \mathbf{X}(s)}{\Delta s} \tag{25}$$

$$(D_s^- \mathbf{X})(s) = \frac{\mathbf{X}(s) - \mathbf{X}(s - \Delta s)}{\Delta s} \tag{26}$$

are difference operators defined along the immersed boundary. Note that since the immersed boundary is a simple closed curve in the model problem considered, it was found to be most convenient to identify the first point, $\mathbf{X}(s_1)$, with the last point, say, $\mathbf{X}(s_{N_B+1})$.

The discretizations of the integrals in (14)–(15) were based on the application of the Trapezoidal Rule for space variables. In what follows, it will be convenient to introduce some simpler notation to alleviate both the indexing on staggered grids and the somewhat cumbersome representation of the discretized interaction equations (23)–(24). First, let the *spreading operation* performed at time t^n in (24) be described by $\mathcal{S}^n$, which will be given by

$$(\mathcal{S}^n \varphi)(\mathbf{x}_{i,j}) = \begin{bmatrix} \Delta s \sum_k \varphi_1(s_k)\, \delta_h^2\left(\left(\mathbf{x}_{i,j} - \frac{h}{2}\mathbf{e}_1\right) - \mathbf{X}_k^n\right) \\ \Delta s \sum_k \varphi_2(s_k)\, \delta_h^2\left(\left(\mathbf{x}_{i,j} - \frac{h}{2}\mathbf{e}_2\right) - \mathbf{X}_k^n\right) \end{bmatrix} \tag{27}$$

for any vector field $\varphi = (\varphi_1, \varphi_2)$ defined on the immersed boundary, where $\mathbf{e}_1$ and $\mathbf{e}_2$ are the unit vectors in the x and y directions, respectively. Note that the convention $\mathbf{X}_k^n = \mathbf{X}^n(s_k)$ was used, where $s_k = s_1 + k\Delta s$, with s_1 some origin arbitrarily chosen for the immersed boundary parametrization.

Similarly, let $\mathcal{S}^{*n}$, denoting the *interpolation operation* performed at time t^n in Eq. (23), be defined by

$$(\mathcal{S}^{*n} \psi)(s_k) = \begin{bmatrix} h^2 \sum_{i,j} \psi_1\left(\mathbf{x}_{i-\frac{1}{2},j}\right)\, \delta_h^2\left(\mathbf{x}_{i-\frac{1}{2},j} - \mathbf{X}_k^n\right) \\ h^2 \sum_{i,j} \psi_2\left(\mathbf{x}_{i,j-\frac{1}{2}}\right)\, \delta_h^2\left(\mathbf{x}_{i,j-\frac{1}{2}} - \mathbf{X}_k^n\right) \end{bmatrix} \tag{28}$$

for any cell-edge vector field $\psi = (\psi_1, \psi_2)$, where ψ_1 and ψ_2 are grid functions defined at the middle of the vertical and horizontal cell edges, respectively.

In terms of the spreading and interpolation operators (27) and (28), Eqs. (23)–(24) can be rewritten as

$$\mathbf{X}_k^{n+1} = \mathbf{X}_k^n + \frac{\Delta t}{2}\left[(\mathcal{S}^{*n}\mathbf{u}^n)(s_k) + (\mathcal{S}^{*n+1}\mathbf{u}^{n+1})(s_k)\right] \tag{29}$$

$$\mathbf{F}_{i,j}^{n+1} = \left[T_0 \mathcal{S}^{n+1}\left(D_s^+ D_s^- \mathbf{X}^{n+1}\right)\right](\mathbf{x}_{i,j}). \tag{30}$$

6. DISCRETIZATION OF THE NONLINEAR ADVECTION TERM

The term $[(\mathbf{u}\cdot\nabla)\mathbf{u}]^{n+1/2}$ appearing in (9) is computed explicitly in time. A quite standard second-order spatial discretization of this term will be applied to a predicted value of the

velocity field at time $t^{n+1/2}$. Following Bell, Colella, and Glaz [5], the predicted velocity field will be obtained by the Taylor expansion

$$\mathbf{u}_{i,j}^{n+\frac{1}{2}} = \mathbf{u}_{i,j}^{n} + \frac{\Delta t}{2}(\mathbf{u}_t)_{i,j}^{n},$$

where the time derivative $\mathbf{u}_t$ can be computed with help from the Navier–Stokes equations. Thus, one obtains

$$\mathbf{u}_{i,j}^{n+\frac{1}{2}} = \mathbf{u}_{i,j}^{n} + \frac{\Delta t}{2}\left\{\frac{\mu}{\rho}L\mathbf{u}_{i,j}^{n} - [(\mathbf{u}\cdot\nabla)\mathbf{u}]_{i,j}^{n} - \frac{Gp_{i,j}^{n}}{\rho} + \frac{\mathbf{F}_{i,j}^{n}}{\rho}\right\}, \tag{31}$$

for the predicted velocity field, which can be rewritten as

$$\frac{\mathbf{u}_{i,j}^{n+\frac{1}{2}} - \mathbf{u}_{i,j}^{n}}{\Delta t/2} + \frac{Gp_{i,j}^{n}}{\rho} = \frac{\mu}{\rho}L\mathbf{u}_{i,j}^{n} - [(\mathbf{u}\cdot\nabla)\mathbf{u}]_{i,j}^{n} + \frac{\mathbf{F}_{i,j}^{n}}{\rho}. \tag{32}$$

Note that besides the velocity at the half time level, the pressure gradient at time n is also not known. By imposing the requirement

$$D\cdot\mathbf{u}^{n+\frac{1}{2}} = 0, \tag{33}$$

one can solve (32)–(33) simultaneously for the velocity field and for the pressure.

Since the vectors $\mathbf{u}$ and $\mathbf{F}$ appearing on the right hand side of (32) are known at time t^n, the computation of the predicted velocity field $\mathbf{u}^{n+1/2}$ depends on the definition of the spatial discretization for the nonlinear advection term at time t^n. A standard second-order discretization is (see [31])

$$\begin{aligned}[(\mathbf{u}\cdot\nabla)\mathbf{u}]_{i,j} \approx \Bigg(&u_{i-\frac{1}{2},j}\left(\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{3}{2},j}}{2h}\right) + \overline{v}_{i-\frac{1}{2},j}\left(\frac{u_{i-\frac{1}{2},j+1} - u_{i-\frac{1}{2},j-1}}{2h}\right),\\ &\overline{u}_{i,j-\frac{1}{2}}\left(\frac{v_{i+1,j-\frac{1}{2}} - v_{i-1,j-\frac{1}{2}}}{2h}\right) + v_{i,j-\frac{1}{2}}\left(\frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{3}{2}}}{2h}\right)\Bigg),\end{aligned} \tag{34}$$

which is not in conservation form, and where the time indices n were suppressed in favor of clarity. In the approximation (34),

$$\begin{aligned}\overline{v}_{i-\frac{1}{2},j} &= \frac{v_{i,j-\frac{1}{2}} + v_{i,j+\frac{1}{2}} + v_{i-1,j+\frac{1}{2}} + v_{i-1,j-\frac{1}{2}}}{4}\\ \overline{u}_{i,j-\frac{1}{2}} &= \frac{u_{i-\frac{1}{2},j} + u_{i-\frac{1}{2},j-1} + u_{i+\frac{1}{2},j-1} + u_{i+\frac{1}{2},j}}{4}.\end{aligned}$$

The numerical solution of the system (32)–(33) is obtained on composite MAC-grids by a discrete projection method, similar to the one described in the next section. Once this system has been solved, the computation of the nonlinear advection term $[(\mathbf{u}\cdot\nabla)\mathbf{u}]^{n+1/2}$ is then completed by again applying (34), this time to $\mathbf{u}^{n+1/2}$.

7. NUMERICAL SCHEME AND SOLUTION ALGORITHM

7.1. *Set of Discretized Equations*

The implicit scheme proposed that (9)–(10), (14)–(16) can be solved numerically through an iterative approach. Given *initial guesses* $Gp^{n+1/2,0}$, $\mathbf{X}^{n+1,0}$, and $\mathbf{F}^{n+1/2,0}$ the new state of the system, $(\mathbf{X}^{n+1}, \mathbf{u}^{n+1})$, is computed by

$$\frac{\mathbf{u}^{*,m} - \mathbf{u}^n}{\Delta t} + \frac{Gp^{n+\frac{1}{2},m-1}}{\rho} = \frac{\mu}{\rho} L\left(\frac{\mathbf{u}^{*,m} - \mathbf{u}^n}{2}\right) - [(\mathbf{u}\cdot\nabla)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\mathbf{F}^{n+\frac{1}{2},m-1}}{\rho} \tag{35}$$

$$\frac{\mathbf{u}^{n+1,m} - \mathbf{u}^n}{\Delta t} + \frac{Gp^{n+\frac{1}{2},m}}{\rho} = \frac{\mathbf{u}^{*,m} - \mathbf{u}^n}{\Delta t} + \frac{Gp^{n+\frac{1}{2},m-1}}{\rho} \tag{36}$$

$$D\cdot\mathbf{u}^{n+1,m} = 0 \tag{37}$$

$$\begin{aligned}[I - \lambda^{n+1,m-1}A](\mathbf{X}^{n+1,m} - \mathbf{X}^{n+1,m-1}) \\ = \mathbf{X}^n + \frac{\Delta t}{2}\mathcal{S}^{*n}\mathbf{u}^n - \left(\mathbf{X}^{n+1,m-1} - \frac{\Delta t}{2}\mathcal{S}^{*n+1,m-1}\mathbf{u}^{n+1,m}\right)\end{aligned} \tag{38}$$

$$\begin{aligned}\mathbf{F}^{n+\frac{1}{2},m} &= \frac{1}{2}(\mathbf{F}^n + \mathbf{F}^{n+1,m}) \\ &= \frac{T_0}{2}\left[\mathcal{S}^n\left(D_s^+ D_s^- \mathbf{X}^n\right) + \mathcal{S}^{n+1,m}\left(D_s^+ D_s^- \mathbf{X}^{n+1,m}\right)\right],\end{aligned} \tag{39}$$

where m, $m \geq 1$, is the iteration number within each time step.

In the discretized form of Navier–Stokes equations (35)–(37), the difference operators G, L, and D are those defined by (19), (20), and (18), respectively. The elastic force distribution, the pressure gradient, and the nonlinear advection term are treated as source terms in (35), the last computed by (34) from a predicted velocity field at $t^{n+1/2}$, as explained in the previous section.

The fluid–boundary interaction equations (38)–(39) provide updates for the immersed boundary position and for the elastic force distribution. Equation (38) is a reformulation of Eq. (29) in terms of a fixed-point iteration, similar to that proposed by Mayo and Peskin [24], in which $\lambda^{n+1,m-1}$ is a diagonal matrix obtained by

$$\lambda^{n+1,m-1} = \mathcal{S}^{*n+1,m-1}\mathcal{S}^{n+1,m-1}\mathbf{1},$$

where $\mathcal{S}$ and $\mathcal{S}^*$ were defined in (27) and (28), $\mathbf{1}(s)$ is a constant function assuming 1 on every immersed boundary point, and A is a notation introduced for the difference operator

$$A = \left(\frac{T_0}{\rho}\Delta t^2\right) D_s^+ D_s^-,$$

with D_s^+ and D_s^- given by (25) and (26).

For the model problem used, the operator $[I - \lambda A]$ in (38) has a periodic–tridiagonal structure. Further motivation for the choice of this operator can be found in the work by Mayo and Peskin [24].

7.2. *Projection Method for Composite MAC-Grids*

In practice, the solution of (35)–(37) is computed in three steps by a projection method, specially designed for composite MAC-grids. First, in the *parabolic step*, one must solve the implicit parabolic equation (35),

$$\frac{\mathbf{u}_{i,j}^{*,m} - \mathbf{u}_{i,j}^{n}}{\Delta t} + \frac{1}{\rho} G p_{i,j}^{n+\frac{1}{2},m-1} = \frac{\mu}{\rho} L\left(\frac{\mathbf{u}_{i,j}^{*,m} + \mathbf{u}_{i,j}^{n}}{2}\right) - [(\mathbf{u}\cdot\nabla)\mathbf{u}]_{i,j}^{n+\frac{1}{2}} + \frac{1}{\rho}\mathbf{F}_{i,j}^{n+\frac{1}{2},m-1},$$

for the provisional velocity $\mathbf{u}_{i,j}^{*,m}$.

Second, in the *elliptic step*, one must solve the pressure Poisson equation

$$D\cdot\left(\frac{1}{\rho} G p_{i,j}^{n+\frac{1}{2},m}\right) = D\cdot\left(\frac{\mathbf{u}_{i,j}^{*,m} - \mathbf{u}_{i,j}^{n}}{\Delta t} + \frac{1}{\rho} G p_{i,j}^{n+\frac{1}{2},m-1}\right) \tag{40}$$

with periodic boundary conditions, which is obtained by imposing the incompressibility constraint (37) to (36). Observe that away from the coarse–fine grid interfaces, the product of the difference operators D and G gives the usual five-point stencil for the Laplacian L.

In the third step, to complete the projection, the provisional velocity field $\mathbf{u}_{i,j}^{*,m}$ is decomposed using the pressure just obtained in the second step,

$$\mathbf{u}_{i,j}^{n+1,m} = \mathbf{u}_{i,j}^{*,m} - \frac{\Delta t}{\rho} G\left(p^{n+\frac{1}{2},m} - p^{n+\frac{1}{2},m-1}\right)_{i,j},$$

giving as a result $\mathbf{u}^{n+1,m}$, a discretely divergence-free vector field defined on the entire composite MAC-grid. This step will be referred to as the *decomposition step*.

The simplest possible way to solve (35) is by a Gauss–Seidel method which requires, on composite MAC-grids, the definition of $\mathbf{u}^{*,m}$ on coarse cell edges underneath fine grid patches. For accuracy, they are defined by *cubic interpolation* of the values in the next finer level above. Employing periodic boundary conditions on level 1 for $\mathbf{u}^{*,m}$, one can rewrite the Gauss–Seidel Method on composite MAC-grids in recursive form as

CMACG-GS(level l)
  if level $l > 1$ **then**
    1. **define** $\mathbf{u}^{*,m}$ on level $l-1$ covered cells by cubic interpolation
    2. **set up** $\mathbf{u}^{*,m}$ on ghost cell edges
    3. **perform** one Gauss–Seidel relaxation
    4. **CMACG-GS** (level $l-1$)
  else
    5. **perform** one Gauss–Seidel relaxation
  end
end CMACG-GS

This procedure is repeated for level l varying from l_{finest} to 1 until the residual is "small" in all the grids, for all levels. Recall that all difference operators are well defined, even on grid borders, since ghost cells were appended to them. Equation (35) is well conditioned, usually with 15 to 20 repetitions needed to drop the residual to 10^{-8} on the composite MAC-grid.

The elliptic step is the most difficult step of all. Equation (40), unlike the preceding equation, cannot be solved numerically by the Gauss–Seidel method since this Poisson equation is poorly conditioned; sometimes, thousands of iterations are needed for this

method to converge. Instead, the multilevel–multigrid method described by Minion in [25] is employed. It solves very efficiently Poisson equations on composite MAC-grids and will be described in the following.

The implementation of a residual-correction multigrid strategy depends essentially on the relaxation on individual grids, on the definition of residual problems, especially at coarse–fine grid interfaces, and on the transfer of information upward and downward between two successive levels. For the moment, the refinement ratio between levels will be supposed to be equal to two. This is not essential but it will make the exposition of the multilevel–multigrid strategy clearer.

The problem to be solved on a composite MAC-grid is

$$D \cdot \left(\frac{1}{\rho} G p\right) = D \cdot (\mathbf{rhs}), \tag{41}$$

where **rhs** is just the right hand side of (36), with periodic boundary conditions. All indices were suppressed for a simpler notation.

The residual–correction problem on a grid at the finest level is defined by

$$D \cdot \left(\frac{1}{\rho} G \psi\right) = r \doteq D \cdot (\mathbf{rhs}) - D \cdot \left(\frac{1}{\rho} G p\right), \tag{42}$$

where r is the residual and ψ the correction.

Given an initial guess ψ^0, relaxation is performed on each individual grid by the Gauss–Seidel Method in a red–black fashion,

$$\psi_{i,j}^{k+1} = \frac{\psi_{i+1,j}^k + \psi_{i-1,j}^k + \psi_{i,j+1}^k + \psi_{i,j-1}^k - h^2 r_{i,j}}{4}, \tag{43}$$

where h is the mesh spacing related to the particular grid considered. Since ghost cells at the grid borders provide boundary values, again the relaxation can be extended to those cells, where (43) can be formally applied.

To define the residual problem on the next coarse level, one must first compute the residual of the residual problem (42) at the finer level,

$$\tilde{r} \doteq r - D \cdot \left(\frac{1}{\rho} G \psi\right) = D \cdot \left(\mathbf{rhs} - \frac{1}{\rho} G (p + \psi)\right), \tag{44}$$

and then, through a *restriction operation*, transfer it downward onto the next coarse level, defining, in this way, the coarse residual problem

$$D \cdot \left(\frac{1}{\rho} G \tilde{\psi}\right) = \mathcal{R}_{l-1}^{l}[\tilde{r}], \tag{45}$$

where $\tilde{\psi}$ is the coarse correction and $\mathcal{R}_{l-1}^{l}$ is the restriction opertor.

On uniform grids, normally $\mathcal{R}_{l-1}^{l}$ is performed by *simple average*, that is,

$$\tilde{r}_{I,J}^{l-1} \doteq \mathcal{R}_{l-1}^{l}[\tilde{r}]_{I,J} = \frac{\tilde{r}_{2i,2j} + \tilde{r}_{2i+1,2j} + \tilde{r}_{2i+1,2j+1} + \tilde{r}_{2i,2j+1}}{4}, \tag{46}$$

which preserves the homogeneous Neumann solvability condition

$$\sum_{i,j} D \cdot (\mathbf{rhs})_{i,j} = 0, \tag{47}$$

present in all the residual problems (and naturally satisfied under periodic boundary conditions).

On composite grids, in order to have (47) satisfied on level 1, special care must be taken at coarse–fine grid interfaces; restriction (46) cannot be performed exactly as written. The residual computation (44) suggests another way of performing the restriction on composite grids which will preserve the solvability condition (47). Instead of simply averaging cell-center values like $\tilde{r}$ onto parent cell centers, the restriction is performed on the cell-edge values $[\mathbf{rhs} - \frac{1}{\rho} G(p + \psi)]$ by simply averaging them onto the parent cell edges they cover (also represented by $\mathcal{R}^l_{l-1}$). This procedure can be viewed as a flux correction step. To complete the definition of the restriction operation for composite grids, on uncovered cells, one computes the residual as in (42).

After reaching the coaresest level, corrections ψ are transferred upward to finer levels through the interpolation operator, which is given by a bilinear interpolation. A V-cycle schedule is used to visit all levels, including physical levels.

The basic algorithm used for the multilevel-multigrid method is

MMM(level l)
if level $l > 1$ **then**
1. **relaxation**
   1.1 set up ψ^l on ghost cell centers
   1.2 relax on residual problem $D \cdot (\frac{1}{\rho} G\psi^l) = r^l$, ν_1 times
2. **flux correction**
   2.1 set up ψ^l on ghost cells
   2.2 correct flux at grid borders: $G\psi^l \leftarrow Gp^l + G\psi^l$
3. **residual computation** on level $l - 1$

$$r^{l-1} \leftarrow \begin{cases} D \cdot \left(\mathbf{rhs}^{l-1} - \frac{1}{\rho} G p^{l-1}\right) & \text{on uncoverd cells} \\ D \cdot \mathcal{R}^l_{l-1}\left[\mathbf{rhs}^l - \frac{1}{\rho} G(p^l + \psi^l)\right] & \text{on coverd cells} \end{cases}$$

4. **MMM**(level $l - 1$)
5. **interpolate** and **add** coarse correction: $\psi^l \leftarrow \psi^l + \mathcal{I}^l_{l-1}[\psi^{l-1}]$
6. **relax** on $D \cdot (\frac{1}{\rho} G\psi^l) = r^l$, ν_2 times
7. **add correction** to solution: $p^l \leftarrow p^l + \psi^l$

else
8. **perform** one V-cycle on residual problem

end
end MMM

For the problem presented here, the procedure above is repeated until the maximum residual in all the grids, for all the levels, is less than 10^{-8}.

More details on the multilevel–multigrid method employed, including details on the interpolation stencils and on the case when the refinement ratio is four, can be found in the works by Minion [25] and by Roma [34].

7.3. *Initial Guesses*

Finally, the initial guess for the pressure gradient at time $t^{n+1/2}$ is given by

$$Gp^{n+\frac{1}{2},0} = \begin{cases} 0, & n = 0, \\ Gp^{n-\frac{1}{2}}, & n \geq 1. \end{cases} \tag{48}$$

and for the elastic force distribution, one can use

$$\mathbf{F}^{n+\frac{1}{2},0} = \frac{1}{2}(\mathbf{F}^n + \mathbf{F}^{n+1,0})$$

$$\mathbf{F}^{n+1,0} = T_0 \mathcal{S}^{n+1,0}\left(D_s^+ D_s^- \mathbf{X}^{n+1,0}\right) \tag{49}$$

$$\mathbf{X}^{n+1,0} = \mathbf{X}^n + \frac{\Delta t}{2}(\mathcal{S}^{*n}\mathbf{u}^n + \mathcal{S}^{*n+1,-1}\mathbf{u}^{n+1,-1}), \tag{50}$$

where $\mathbf{X}^{n+1,-1}$ and $\mathbf{u}^{n+1,-1}$ are obtained through the expressions

$$\mathbf{X}^{n+1,-1} = \mathbf{X}^n + \Delta t \mathcal{S}^n \mathbf{u}^n$$

$$\mathbf{u}^{n+1,-1} = \mathbf{u}^n + \Delta t \mathbf{u}_t^n = \mathbf{u}^n + \Delta t \left(\frac{\mu}{\rho} L\mathbf{u}^n - [(\mathbf{u}\cdot\nabla)\mathbf{u}]^n + \frac{\mathbf{F}^n}{\rho} - \frac{Gp^{n-\frac{1}{2}}}{\rho} \right).$$

8. RESULTS

This section presents the numerical results obtained in computations performed with the adaptive version of the Immersed Boundary Method, applied to the model problem described in Section 2. Briefly, the immersed boundary is given by a simple closed curve, the two-dimensional analog of an elastic spherical balloon, filled with the same fluid present outside. Its motion is driven by the elastic force acting on its wall (8), for which the non-negative constant T_0 was taken as 1.0 dyn/cm^2.

At time $t = 0.0$, the initial position of the immersed boundary was an ellipse aligned in the coordinate directions with horizontal semi-axis $a = 0.28125$ and vertical semi-axis $b = 0.75 \times a$, centered at (0.5, 0.5); initially, the fluid was at rest, that is, $\mathbf{u}(\mathbf{x}, 0) = 0.0$, $\mathbf{x} \in \Omega = [0, 1] \times [0, 1]$, Ω doubly periodic. Under these conditions, if the integration time were long enough, the immersed boundary would tend to a circle, its equilibrium configuration, in a damped oscillatory motion.

In all the computations that follow, the mass density ρ has been set to 1.0 g/cm^3, the viscosity μ to 0.01 g/(cm s), and the immersed boundary, as mentioned previously, has been uniformly covered by grids at the finest level. Taking $L_B = 2\pi\sqrt{(a^2 + b^2)/2}$ as the length of the immersed boundary at time $t = 0.0$, the number of immersed boundary points was set to

$$N_B = 2\frac{L_B}{h_{\text{finest}}}, \tag{51}$$

which gives an average density of two immersed boundary points per meshwidth. This density was kept constant in all the runs, whether uniform or composite grids were used in the simulations, with the total number of immersed boundary points adjusted appropriately, and with their location the same for grids with the same finest meshwidth.

With the initial conditions and physical parameters selected as above, the immersed boundary position was integrated up to time $t = 0.5$, when its first oscillation was almost complete. During this motion, the minimum length assumed by the major axis was about 75% of its initial length. The time steps were given by the hyperbolic restriction

$$\Delta t = C \frac{h_{\text{finest}}}{\|\mathbf{u}\|_\infty}, \tag{52}$$

with constant $C = 0.5$.

Before any numerical results for the state variables $(\mathbf{X}, \mathbf{u})$ are presented, a description of how norms on uniform and on composite grids were computed will be given. For the immersed boundary $\mathbf{X}(s_k) = (X_1(s_k), X_2(s_k))$, $1 \le k \le N_B$, the discretized L_2-norm is given by

$$\|\mathbf{X}\|_2 = \left(\sum_{k=1}^{N_B} \left(X_1^2(s_k) + X_2^2(s_k) \right) \Delta s \right)^{1/2},$$

where $\Delta s = h_{\text{finest}}/2$ is approximately the distance between two consecutive immersed boundary points.

For the fluid velocity, there are two different discretizations of the domain Ω on a staggered grid: Ω_u^h, where the first component is defined, is the union of all midpoints of vertical edges, and Ω_v^h, where the second component is defined, is the union of all midpoints of the horizontal edges. Bearing that in mind, one sees that the L_2-norm of the fluid velocity is given by

$$\|\mathbf{u}\|_2 = \left(\sum_{i \in \Omega_u^h} u_i^2 a_i + \sum_{j \in \Omega_v^h} v_j^2 b_j \right)^{1/2},$$

where

$$a_i = \begin{cases} H^2, & \text{on the coarse cell edges,} \\ H^2/r^2, & \text{on the fine cell edges,} \\ H^2(r+1)/(2r^2), & \text{on the coarse–fine interface edges,} \end{cases} \tag{53}$$

with r the refinement ratio between levels l and $l+1$, and H the mesh spacing of level l. The weights b_j are defined in the same way.

The first series of numerical results, displayed in Table I, were obtained by applying the iterative method (35)–(39) on four $N \times N$ uniform grids, $N = 16$, 32, 64, and 128, and then

TABLE I
Uniform Grid Results

h_{finest}	1/16		1/32		1/64		1/128
N	16	Ratio	32	Ratio	64	Ratio	128
$\|\mathbf{X}_N - \mathbf{X}_{256}\|_2$	1.166×10^{-2}	8.61	1.355×10^{-3}	2.16	6.265×10^{-4}	3.19	1.961×10^{-4}
$\|\mathbf{u}_N - \mathbf{u}_{256}\|_2$	1.300×10^{-1}	2.69	4.833×10^{-2}	3.09	1.564×10^{-2}	3.41	4.582×10^{-3}

by comparing the immersed boundary position and the fluid velocity at the final time $t = 0.5$ with a finer solution obtained on the uniform grid with $N = 256$.

Further remarks on the norm computation must be made at this point. From one test case to the next finer one, the set of immersed boundary points was increased by including a new point midway between each two previously existing points. By adopting this procedure, the density of immersed boundary points per meshwidth was kept constant from one case to the next finer one, and the norms involving the immersed boundary points could be computed simply by using the points which were "common" to both, the case being run and the 256×256 case (these common points occupied the same position at the initial time; the other points required on the finer grid were simply neglected). Also, note that the fluid velocity is computed at different points for grids with different meshwidths; so, before computing the norms of their differences, the solution defined on the finer grid had to be interpolated to the coarser grid points (second-order accurate interpolation was used).

For each time step, the method was allowed to iterate until the difference between two consecutive updates of the immersed boundary position at the end of that time step satisfied a certain prespecified convergence criterion which, kept fixed for all the cases, was chosen as

$$\|\mathbf{X}^{n+1,m} - \mathbf{X}^{n+1,m-1}\|_\infty \le (0.5 \times 10^{-4}) h_{256}, \tag{54}$$

where n denotes the current time step, m the current iteration number (within the particular time step n), and h_{256} the meshwidth of the 256×256 uniform grid.

Since the comparison of the results was made with a solution which was not as fine as ideally it should be, one cannot expect convergence ratios of two and four to mean first- and second-order accuracy, respectively. Instead, assuming that the numerical solution of the immersed boundary position has an asymptotic expansion in powers of h, the accuracy q of the method can be estimated, for example, from the ratio

$$\frac{\|\mathbf{X}_{016} - \mathbf{X}_{256}\|_2}{\|\mathbf{X}_{032} - \mathbf{X}_{256}\|_2} \approx \frac{\left\|\left(\frac{1}{16}\right)^q \mathbf{E}_q - \left(\frac{1}{256}\right)^q \mathbf{E}_q\right\|_2}{\left\|\left(\frac{1}{32}\right)^q \mathbf{E}_q - \left(\frac{1}{256}\right)^q \mathbf{E}_q\right\|_2} = \left(\frac{2^{4q} - 1}{2^{3q} - 1}\right), \tag{55}$$

where $\mathbf{E}_q$ is a coefficient which may depend on time but not on h. Estimates can be similarly derived from the other uniform grid results, giving

$$\frac{\|\mathbf{X}_{032} - \mathbf{X}_{256}\|_2}{\|\mathbf{X}_{064} - \mathbf{X}_{256}\|_2} \approx \left(\frac{2^{3q} - 1}{2^{2q} - 1}\right) \tag{56}$$

$$\frac{\|\mathbf{X}_{064} - \mathbf{X}_{256}\|_2}{\|\mathbf{X}_{128} - \mathbf{X}_{256}\|_2} \approx \left(\frac{2^{2q} - 1}{2^q - 1}\right). \tag{57}$$

Note that the same estimates will hold if similar conditions are assumed for the fluid velocity.

Using expressions (55)–(57), one can obtain estimates for the convergence ratios for $q = 1$ and $q = 2$, and then compare them with the computed convergence ratios contained in Table I. Table II shows these estimated convergence ratios for first- and second-order accurate methods. A comparison between the computed convergence ratios, displayed in Table I, and the estimated convergence ratios, displayed in Table II, shows that the method exhibits a first-order asymptotic behavior for both state variables in the L_2-norm.

TABLE II
Estimated Convergence Ratios for First-Order ($q = 1$) and Second-Order ($q = 2$) Accurate Methods

Order	Ratio (16–32)	Ratio (32–64)	Ratio (64–128)
$q = 1$	2.14	2.33	3.00
$q = 2$	4.05	4.20	5.00

The next series of numerical results were obtained by employing the adaptive multilevel approach to the same model problem. Composite grids were generated by flagging a seven by seven coarse-cell neighborhood around each of the immersed boundary points. This avoided too many regridding steps by making sure that the boundary points would have enough room to move for several time steps. Also, the border of a fine level was maintained three coarse cells away from the border of the next coarser level (a more stringent criterion than would be sufficient for grids to be properly nested). These numbers were selected after few numerical experiments and may be case dependent. By following this procedure, the cells so flagged served as input to the grid generation routines which, with at least 75% of efficiency, produced the fine grids.[1]

On composite grids generated as explained above, the regridding process took place only when an immersed boundary point moved fewer than four cells away from the border of the finest level. This guaranteed that there would be at least two cells between the border of the finest level and the support of the discretized delta function, a precautionary measure taken to prevent errors in the fluid velocity at coarse–fine interfaces from affecting the computation of the immersed boundary velocity during the interpolation step. Note that if discretizations with wider supports were used for the delta function, the equivalent regridding criterion would require more than four cells. For example, if a four-cell supported delta function were used, regridding would have to take place when an immersed boundary point moved fewer than five cells away from the border of the finest level to keep this border at least two cells away from the delta function support.

In the first two numerical columns of Table III, one extra level refined by the ratio 2 was added to base levels of 32×32 and 64×64 cells. Briefly, the resulting composite grids will be denoted respectively by "32(L2R2)" and "64(L2R2)," whose reading, for example for the first case, is "32 cells in each direction in the base level, two levels altogether, refinement ratio 2." The degree of accuracy obtained was comparable to that obtained for the 64×64 and 128×128 uniform grids respectively (compare with the last two columns of Table I).

Note that only grids at the same level were used to cover the immersed boundary, allowing for its norms to be computed in the same way in which they were computed on the uniform grids. For the norms involving the fluid velocity, the solution obtained on the uniform 256×256 grid had to be first interpolated to the composite grid points under consideration.

The last two columns of Table III illustrate in more depth the potential of this adaptive multilevel approach. Due to the localized refinement around the immersed boundary, results comparable to those obtained on a uniform grid 128×128 were obtained on the composite

[1] Efficiency was measured by the ratio between the area of the input tagged region over the area of the output generated grid.

TABLE III
Composite Grid Results

h_{finest}	1/64	1/128	1/128	1/128
N	32(L2R2)	64(L2R2)	32(L3R2)	32(L2R4)
$\|\mathbf{X}_N - \mathbf{X}_{256}\|_2$	6.186×10^{-4}	2.016×10^{-4}	1.951×10^{-4}	1.951×10^{-4}
$\|\mathbf{u}_N - \mathbf{u}_{256}\|_2$	1.569×10^{-2}	4.598×10^{-3}	4.637×10^{-3}	4.640×10^{-3}

grids 32(L3R2) and 32(L2R4). In all three cases, the mesh spacing around the immersed boundary was the same, equal to 1/128. Figure 4 shows all the composite grids used in the case 32(L3R2).

Compared to the original regridding strategy employed in [34], when regridding was performed according to a fixed schedule every few time steps, the current strategy can generate up to less than 10% of the total number of composite grids needed previously. The main reason for such improvement is that, now, the total number depends essentially on the amplitude of the immersed boundary motion. Note that if the code is to adapt based on a nonsmooth flow field elsewhere, and not just around the immersed boundary, it may be necessary to employ both the current and the previous strategies together, the final decision being problem dependent. Nevertheless, when the Reynolds number is not so high (500 or so, e.g., the blood flow in the heart chambers), the flow is not expected to be nonsmooth away from the immersed boundary; the natural choice for the regridding strategy, in this

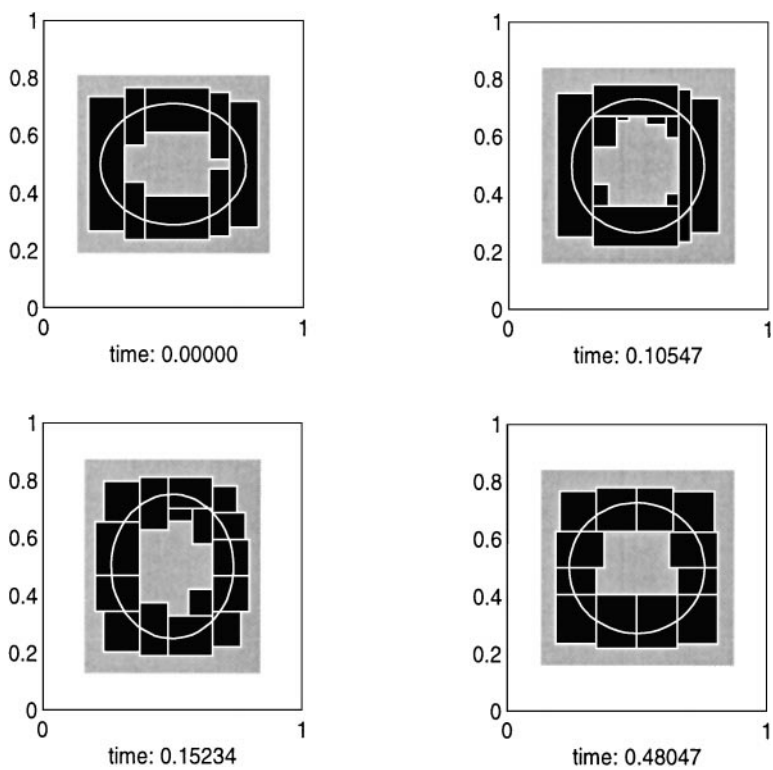

FIG. 4. Composite grids and boundary positions used for case 32(L3R2).

case, is the one introduced here. Less than 0.5% of the total running time was spent to recompute and move the composite grids.

Although composite grids and uniform grids with the same mesh spacing around the immersed boundary exhibited comparable resolutions, a considerably smaller number of state variables had to be defined for composite grids. For example, the solution of the problem on the uniform grid 128×128 required the definition of the state variables on 16,384 cells, while an equivalent resolution obtained on the composite grid 32(L3R2) required only about 45% of that number of cells, distributed among about 10 grid patches (not taking into account ghost cells).

The number of computational cells plays an important role. By requiring fewer computational cells than the uniform grid Immersed Boundary Method, this adaptive approach can be used to solve problems with resolutions which could not be achieved previously on uniform meshes due to the prohibitive amount of computer memory required.

For problems that can be tackled by both approaches, where the amount of memory is not the issue, the smaller number of computational cells needed by the adaptive approach suggests that computer time might be saved for, in this case, the solution of smaller systems will be required. Since the current adaptive code has not yet been optimized for performance, it remains to be seen whether savings in running time can in fact be achieved by the methodology described in this paper.

9. CONCLUSION

By combining the Immersed Boundary Method with an adaptive mesh refinement, one obtains a multilevel version of that method with self-adaptive capabilities.

This approach is tested for a particular two-dimensional model problem, for which *no significant difference* is found between the solutions obtained on a mesh refined locally around the immersed boundary, and on the associated uniform mesh, built with the resolution of the *finest level*.

A Crank–Nicholson type of scheme is the basis for the second-order projection method employed to solve the Navier–Stokes equations on composite MAC-grids. The nonlinear convection term is explicitly computed at half-time levels with simple edge-centered difference stencils, suited for Reynolds numbers in the range of 10 to 100. A rather sophisticated multilevel–multigrid method is used to solve numerically the pressure Poisson equation in the projection step.

An implicit version of the Immersed Boundary Method is employed to free the method from its time-step restriction

$$\Delta t = O\left(h_{\text{finest}}^2\right),$$

which can be intolerable if several levels are used. The elastic force distribution was introduced as a forcing term in the parabolic step and the immersed boundary position was updated by the Trapezoidal Rule.

Although formally second-order accurate, in practice, the method is only first-order accurate (overall accuracy) due to the nonsmooth flow field near the immersed boundary. Another numerical aspect specific to this implementation of the Immersed Boundary Method is that it employs a discretization of the delta function which is only three-cell supported.

The new regridding strategy made the total number of composite grids generated dependent on the amplitude of the immersed boundary motion. Compared to the original regridding strategy employed in [34], when regridding was performed according to a fixed schedule every few time steps, the current strategy can generate up to less than 10% of the total number of composite grids needed previously. For applications where the Reynolds number is not so high, the flow is expected to be smooth away from the immersed boundary, and the natural choice for the regridding strategy is the one introduced here.

ACKNOWLEDGMENTS

The authors are in debt to David McQueen, Louis Howell, Michael Minion, Nathaniel Cowen, Philip Colella, Randall LeVeque, and Zhilin Li for their valuable suggestions and important insights. This work was supported in part by the research grants FAPESP 89/2626-3, Brazil, and by NSF BIR-9302545, DOE DE-FG02-92ER25139, AFORS F49620-97-1-0322, and an NYU Dean's Dissertation Fellowship, U.S.A.

REFERENCES

1. G. Agresar, *A Computational Environment for the Study of Circulating Cell Mechanics and Adhesion*, Ph.D. thesis, The University of Michigan (1996).
2. G. Agresar, J. J. Linderman, G. Tryggvason, and K.G. Powell, An adaptive, cartesian, front-tracking method for the motion, deformation and adhesion of circulating cells, *J. Comput. Phys.* **143**, 346 (1998).
3. S. A. Bayyuk, K. G. Powell, and B. van Leer, A simulation technique for 2-D unsteady inviscid flows around arbitrary moving and deforming bodies of arbitrary geometry, in *Proceedings, AIAA 11th Computational Fluid Dynamics Conference, Orlando, FL., July 6–9, 1993*, pp. 1013–1024.
4. J. Bell and D. L. Marcus, A second-order projection method for variable density flows, *J. Comput. Phys*. **101**, 334 (1992).
5. J. Bell, P. Colella, and H. M. Glaz, A second-order projection method for the incompressible Navier–Stokes equations, *J. Comput. Phys*. **85**, 257 (1989).
6. M. J. Berger, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations*, Ph.D. thesis, Stanford University (1982).
7. M. J. Berger and P. Colella, Local adaptive mesh refinement for shock hydrodynamics, *J. Comput. Phys*. **82**, 64 (1989).
8. M. J. Berger and J. Oliger, Adaptive mesh refinement for hyperbolic partical differential equations, *J. Comput. Phys*. **53**, 484 (1984).
9. M. J. Berger and I. Rigoutsos, An algorithm for point clustering and grid generation, *IEEE Trans. Systems, Man, Cybernet*. **21**(5), 1278 (September/October 1991).
10. R. P. Beyer, A computational model of the cochlea using the immersed boundary method, *J. Comput. Phys*. **98**, 145 (1992).
11. A. J. Chorin, Numerical solution of the Navier–Stokes equations, *Math. Comp*. **22**(104), 745 (1968).
12. A. J. Chorin, On the convergence of discrete approximations to the Navier–Stokes equations, *Comm. Pure Appl. Math*. **23**(106), 341 (1969).
13. P. Colella, A multidimensional second order godunov scheme for conservation laws, Technical Report LBL-17023, Lawrence Berkeley Laboratory (May 1984). [Unpublished]
14. L. J. Fauci, Interaction of oscillating filaments—A computational study, *J. Comput. Phys*. **86**, 294 (1990).
15. A. L. Fogelson, A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting, *J. Comput. Phys*. **56**, 111 (1984).
16. J. A. Greenough, V. Beckner, R. B. Pember, W. Y. Crutchfield, J. B. Bell, and P. Colella, An adaptive multifluid interface-capturing method for compressible flow in complex geometries, in *Proceedings, AIAA 26th Computational Fluid Dynamics Conference, San Diego, 1995*; AIAA Paper 95-1718.

17. H. Haj-Hariri, Q. Shi, and A. Borhan, Thermocapillary motion of deformable drops at finite Reynolds and Marangoni numbers, *Phys. Fluids* **9**(4), 845 (1997).
18. F. H. Harlow and J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluids with free surfaces, *Phys. Fluids* **8**(12), 251 (1965).
19. L. H. Howell and J. B. Bell, An adaptive mesh projection method for viscous incompressible flow, *SIAM J. Sci. Comput*. **18**(4), 996 (July 1997).
20. R. J. LeVeque and Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM J. Numer. Anal*. **31**(4), 1019 (August 1994).
21. R. J. LeVeque and Z. Li, *Immersed Interface Methods for Stokes Flow with Elastic Boundaries or Surface Tension*, Technical Report 95-01, University of Washington (February 1995).
22. Z. Li, *The Immersed Interface Method—A Numerical Approach for Partial Differential Equations with Interfaces*, Ph.D. thesis, University of Washington (1994).
23. Z. Li, *Immersed Interface Method for Moving Interface Problems*, Technical Report CAM 95-25, University of California, Los Angeles, CA 90024-1555 (May 1995).
24. A. A. Mayo and C. S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, *Contemp. Math*. **141**, 261 (1993).
25. M. L. Minion, *Two Methods for the Study of Vortex Patch Evolution on Locally Refined Grids*. Ph.D. thesis, Lawrence Berkeley Laboratory—University of California, Berkeley, 94720 (May 1994).
26. C. S. Peskin, *Flow Patterns around Heart Valves: A Digital Computer Method for Solving the Equations of Motion*, Ph.D. thesis, Albert Einstein College of Medicine—Yeshiva University (July 1972). [University Microfilms #72-30, 378]
27. C. S. Peskin, Flow patterns around heart valves: A numerical method, *J. Comput. Phys*. **10**, 252 (1972).
28. C. S. Peskin and D. M. McQueen, A three-dimensional computational method for blood flow in the heart. I. Immersed elastic fibers in a viscous incompressible fluid, *J. Comput. Phys*. **81**, 372 (1989).
29. C. S. Peskin and D. M. McQueen, Computational biofluid dynamics, *Contemp. Math*. **141**, 161 (1993).
30. C. S. Peskin and D. M. McQueen, A general method for the computer simulation of biological systems interacting with fluids, in *Proceedings, SEB Symposium on Biological Fluid Dynamics, Leeds, England, July 5–8, 1994*.
31. R. Peyret and T. D. Taylor, *Computational Methods for Fluid Flow* (Springer-Verlag, Berlin/New York, 1990), 3rd ed.
32. B. F. Printz, *Computer Modeling of Blood Flow through the Heart during the Complete Cardiac Cycle*, Ph.D. thesis, City University of New York (1992).
33. J. J. Quirk, *An Adaptive Mesh Refinement Algorithm for Computational Shock Hydrodynamics*, Ph.D. thesis, Cranfield Institute of Technology, UK (1991).
34. A. M. Roma, *A Multilevel Self-Adaptive Version of the Immersed Boundary Method*, Ph.D. thesis, Courant Institute of Mathematical Sciences—New York University (January 1996). [University Microfilms #9621828]
35. M. E. Rosar, *A Three-Dimensional Computer Model for Fluid Flow through a Collapsible Tube*, Ph.D. thesis, Courant Institute of Mathematical Sciences—New York University (June 1994).
36. M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, An adaptive level set approach for incompressible two-phase flows, in *Proceedings, the ASME Fluids Engineering Summer Meeting: Forum on Advances in Numerical Modeling of Free Surface and Interface Dynamics, 1996*, Vol. 3, p. 355.
37. M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, An adaptive level set approach for incompressible two-phase flows, *J. Comput. Phys*., to appear.
38. C. Tu and C. S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: A comparison of three methods, *SIAM J. Sci. Statist. Comput*. **13**, 1361 (1992).
39. S. O. Unverdi and G. Tryggvason, A front-tracking method for viscous, incompressible flows, *J. Comput. Phys*. **100**, 25 (1992).